Instructions.

1. Solution may not be submitted by students in pairs.

2. You may submit a pdf of the homework, either printed or hand-written and scanned, as long as it is easily readable.

3. If your solution is illegible not clearly written, it might not be graded.

4. Unless otherwise stated, you should prove the correctness of your answer. A correct answer without justification may be worth less.

5. If you have discussed any problems with other students, mention their names clearly on the homework. These discussions are not forbidden and are actually encouraged. However, you must write your whole solution yourself.

6. Unless otherwise specified, all questions have the same weight.

7. You may refer to data structures or their properties studied in class without having to repeat details, and may reference theorems we have studied without proof. If your answer requires only modifications to one of the algorithms, it is enough to mention the required modifications, and the effect (if any) on the running time and on other operations that the algorithm performs.

8. In general, a complete solution should contain the following parts:

   (a) A high level description of the data structures (if needed). E.g. We use a binary balanced search tree. Each node contains, a key and pointers to its children. We augment the tree so each node also contains a field...

   (b) A clear description of the main ideas of the algorithm. You may include pseudocode in your solution, but this may not be necessary if your description is clear.

   (c) Proof of correctness (e.g. show that your algorithm always terminates with the desired output).

   (d) A claim about the running time, and a proof showing this claim.
1. Explain how you would use hash functions to find if your computer contains two identical files (possibly under different names). Give a pseudocode of your solution. Specify which and how your hash functions are used. Do not use values provided by the file system (such as MD5). Your algorithm should be as efficient as possible.

2. Consider a set of files $F = \{f_1 \ldots f_n\}$. Each with at most 10KB. Different files might have different length.
   
   (a) We treat each file as a key. Is there a perfect hash function for these files that would be appropriate to be used with a hash table with size $m = O(n)$?
   
   (b) If yes, what is the expected running time of the algorithm that finds this function? Prove your answer.
   
   (c) Assume $m = n^3$. We pick at random a function $h$ from a universal family. For two files $x, y \in F$ we define a random variable $c_{xy}(h)$ which is 1 if $h(x) = h(y)$ and is 0 if $h(x) \neq h(y)$. What is $E(c_{xy}(h))$?

3. Let $S = \{x_1 \ldots x_n\}$ be a set of points on the $x$-axis, given sorted in increasing order. Suggest an algorithm that in $O(n)$ time finds a point $c$ that minimizes

   $$\sum_{x_i \in S} |x_i - c|$$

   Note that the running time is worst-case.

4. Let $S = \{h_1 \ldots h_n\}$ be a set of halfplanes in 2D, and let $\ell$ be a line. Let $c = c_1 x + c_2 y$ be a cost function (so you are given the parameters $c_1$ and $c_2$). Show an algorithm that in $O(n)$ (worst-case) time finds the point on $\ell$ that minimizes the cost function, and is inside all halfplanes.
   
   This problem refer to in the slides as the 1-dimensional Linear-Programming, or 1DLP. All notations are as appeared in the slides.

5. You are given a set $B$ of $n$ points in the plane, and a set $R$ of $n$ points in the plane. Each point is given by its coordinates. Suggest an ILP formalization for determining if there is a 1-1 matching between them, where each red point is match to a blue point whose distance is at most 1 unit away.

6. You are given a set $S = \{\ell_1 \ldots \ell_n\}$ of non-vertical lines in the plane. Suggest an algorithm with expected running time $O(n)$ that finds the shortest vertical line segment that crosses every line of $S$.
   
   Hint-express this problem as a Linear Programming problem in in low dimension.

7. You are given a set of $R$ of red points an a set $B$ of blue points. Suggest an algorithm that in expected time $O(n)$ finds whether there is a plane $h$ such that in one of its sides there are only blue points, and on its other side only red ones.
   
   You could assume that no 3 input points lie on the same line.
8. Let $G(V, E)$ be a graph (not necessarily bipartite) and assume we are also given for every edge $e_j \in E$ a weight $w(e_j)$ which is a positive real number. We define the max-weight matching problem as follows:

Find a subset $F \subseteq E$ such that every vertex of $v$ is on at most one edge of $F$, and in addition $\sum_{e \in F} w(e)$ is as large as possible. Show an ILP formalization of this problem.

9. Under the same conditions of the previous questions, find a subset $F \subseteq E$ such that every vertex of $v$ is on exactly one edge of $F$ and in addition $\sum_{e \in F} w(e)$ is as small as possible. This problem is known as the min-weight perfect matching.

10. Show an example of a linear programming problem in the (two dimensional $xy$-plane), that has solution very close (but not including) the point $(0, 0)$, but the only integer solution has $x$-value $\geq 1000$. Here the cost function is to minimize $x$. That is, find the point $(x, y)$ with the smallest minimum value.